Vers l'intégration dynamique de contrats dans des architectures orientées services : une experience applicative du modèle au code

Sébastien Mosser, Mireille Blay-Fornarino, Philippe Collet, Philippe Lahire

Vers l'intégration dynamique de contrats dans des architectures orientées services : une expérience applicative du modèle au code

Sébastien Mosser*, Mireille Blay-Fornarino*
Philippe Collet **, Philippe Lahire**

I3S, Université de Nice-Sophia Antipolis, CNRS,
930 route des colles, , BP145, 06903 Sophia Antipolis Cedex
*{mosser,blay}@polytech.unice.fr,
**{Philippe.Collet,Philippe.Lahire}@unice.fr,
http://rainbow.i3s.unice.fr

Résumé. La flexibibilité offerte par les nouvelles architectures orientées services (SOA) renforce les besoins en contractualisation des services fournis et usages de ces services. L'expression de ces exigences et contrats se situe alors à tous les niveaux du cycle de vie des applications et nécessite des mises en oeuvre différentes en fonction de leur nature et des plates-formes ciblées. Dans le cadre du RNTL FAROS, nous avons mené une première expérimentation pour l'introduction de contrats dans une application basée sur une architecture SOA. Nous présentons ici la démarche, les mises en oeuvre et discutons les avantages attendus de l'approche dirigée par les modèles pour gérer les évolutions de l'application.

1 Introduction

Afin de faciliter la mise en place et la maintenance de très grands systèmes d'information évolutifs, les architectures orientées services reposent sur un découpage des applications en des services indépendants, faiblement couplés (MacKenzie et al., 2006). La construction de services résultant de la composition de plusieurs services est alors explicitée sous la forme d'orchestrations, dont le résultat est un nouveau service. La simplicité de publication des services et leur usage facilite ainsi le développement d'applications reposant sur la composition de services.

La production de nombreux services disponibles et les nouvelles techniques de programmation permettent aujourd'hui la mise en place de composition de services qui évoluent en fonction du contexte d'exécution. Le défaut de service ou l'usage malveillant d'un autre peuvent alors occasionner des problèmes dans les applications résultantes des compositions. Il est donc particulièrement utile de définir clairement les propriétés respectées par un service et les conditions à respecter pour utiliser et composer ces services. Lors de l'usage d'un service, il y a donc établissement d'un contrat tacite ou explicite entre les fournisseurs et consommateurs de services(Ludwig et al., 2004). Les services et les contextes d'exécution évoluant, les clauses de

ces contrats doivent également évoluer. Les contrats peuvent être eux-même à l'initiative des évolutions des applications lorsqu'ils sont violés en modifiant par exemple les assemblages. Ceci est d'autant plus important que les architectures orientées services évoluent actuellement vers des approches mixtes entre composants et services (Open SOA, 2007; Zou et Duan, 2006; Collet et al., 2007a). Il est donc nécessaire d'autoriser une intégration dynamique des contrats qui prenne en charge les modifications d'assemblages et les compositions des contrats.

La technologie des Web Services offre un support particulièrement intéressant à la mise en oeuvre des architectures orientées services. En effet, au cœur de l'approche Web Services, les objectifs sont l'interopérabilité supportée par un protocole de communication et la standardisation des services (fichiers WSDL). Cependant, l'intégration des contrats sur les Web Services n'a pas été à ce jour standardisé et reste dépendante des infrastructures de supports (Ludwig et al., 2004; Lazovik et al., 2004; Baresi et al., 2004; Baresi et Guinea, 2005). La vérification d'un contrat à l'exécution repose alors sur un déploiement de codes techniques, dépendant de la nature du contrats et des constituants mis en jeux. De tels déploiements exigent une expertise technique importante fortement dépendante de la plate-forme cible. La multiplicité des plates-formes et leur évolution sont alors un frein à l'introduction et au maintien des contrats dans les applications. Pour tenir compte de ce besoin de forte évolutivité, tout en conservant l'interopérabilité des architectures à base de Web Services, l'ingénierie des modèles (IDM) apporte des éléments de réponses en basant le développement logiciel sur la définition de modèles faisant abstraction des infrastructures techniques. Ces modèles doivent être conformes à des métamodèles pour lesquels des outils de transformations ont été définis, permettant d'adresser à partir d'une modélisation la génération des codes vers de multiples plates-formes. Dans cet article, nous présentons un exemple d'utilisation de l'IDM pour l'intégration de contrats dans une application existante, et montrons comment l'IDM peut intervenir à l'exécution pour assurer, indépendamment de la plate-forme ciblée, la composition des contrats. Ce travail a été réalisé dans le cadre du projet RNTL FAROS[1].

Afin d'illustrer notre propos, nous commençons par présenter une partie de l'application SEDUITE (cf.2), un système de diffusion d'informations actuellement en exploitation, et montrons sur un scénario simple les besoins en terme de contrôle et de modification des assemblages de services pour réagir aux changements de contexte d'exécution. Dans la partie 3, nous établissons l'existence de plates-formes ayant ces capacités à définir des éléments contractuels et à modifier dynamiquement les assemblages de services. Nous présentons brièvement la mise en œuvre dans la plate-forme ADORE/CoCoNet qui sert de support à l'étude relatée dans cet article. Nous explicitons ensuite la démarche IDM visée par le projet FAROS au travers de notre exemple (cf. 4). Nous distinguons en particulier les éléments pris en charge par la plate-forme de ceux qui devraient être pris en charge au niveau d'une modélisation de l'application. Enfin nous concluons cet article en présentant les perspectives de ces travaux.

2 Etude de cas : SEDUITE

2.1 Un service de diffusion des informations

L'application SEDUITE[2] (Blay-Fornarino et al., 2007) vise à fournir un Système d'Information (SI) pour les établissement scolaires, permettant la diffusion d'informations sur différents

[1] Les livrables du projet cités dans cet article sont disponibles à l'adresse `http://www2.lifl.fr/faros/Main/Livrables`.

[2] *SErvices de Diffusion Ubiquitaire d'InformaTions dans des Établissements scolaires*

supports inhérents à l'établissement (informatique ambiante). Cette application est en exploitation au sein de l'Ecole Polytechnique Universitaire Sophia ainsi qu'à l'Institut Education Sensorielle *Clément Ader* à Nice.

Dans le cadre de cette étude, nous nous intéressons au noyau fonctionnel de l'application et pas aux supports de diffusion (écrans fixes, supports mobiles, ...). Développé selon les règles des Architectures Orientées Services (MacKenzie et al., 2006), ce noyau est composé d'un ensemble de Web Service dits *sources* (fournisseurs d'informations) et de services *composites*, résultant d'orchestration agrégeant les différentes sources d'informations disponibles et utiles.

Cette application est développée sur les plates-formes CoCoNet & Adore (Balligand et al., 2007) qui, elles-mêmes, reposent sur `C#/.Net` (cf. section 3).

Sources d'informations Nous nous focalisons dans le cadre de cet article à une version simplifiée du système d'information Seduite en nous limitant à trois sources d'informations :

- `NewsProvider` fournit des informations relatives au fonctionnement général de l'établissement (arrivée d'invité, report de cours, ...) ;
- `Timetable` fournit des informations relatives aux emplois du temps de l'école ;
- `WebCal` est relié aux emplois du temps de certains enseignants de l'établissement.

Dans notre scénario,le Web Service de gestion des emplois du temps des enseignants n'est pas déployé au temps T_0.

Services techniques En plus des précédents services, qualifiés de *«métiers»*, le système d'information repose sur différents services *«techniques»* permettant l'interaction avec les plates-formes utilisées pour sa gestion. On notera par exemple l'`OrchestrationServer` qui permet le déploiement dynamique d'orchestration dans l'application (via Adore), et le `CoCoNetManager` qui permet la pose et le retrait des contrats sur les services déployés.

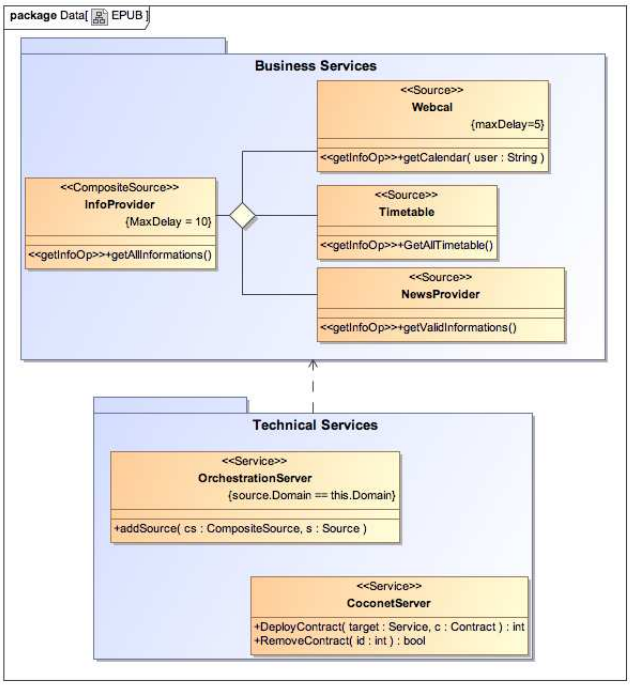
Fonction d'orchestration Afin de respecter les propriétés de couplage faible induites par les Soa, les services ne sont pas directement interconnectés. Des fonctions d'orchestration définissent des services qui agrègent les services sources d'information et/ou services techniques.

Au temps T_0, une fonction d'orchestration expose l'opération `GetAllInformations` qui agrège les informations en provenance des services `NewsProvider` et `Timetable`. Elle est visible pour les utilisateurs comme un service `InformationProvider`. La figure 1(b) visualise dans le formalisme Bpmn[3](White, 2006) cette fonction d'orchestration initiale, tandis que la figure 1(a) donne une vision des services utilisés dans l'application exprimée dans un diagramme de classes UML. Ces modélisations sont obtenues par rétro-ingénierie de l'application en production.

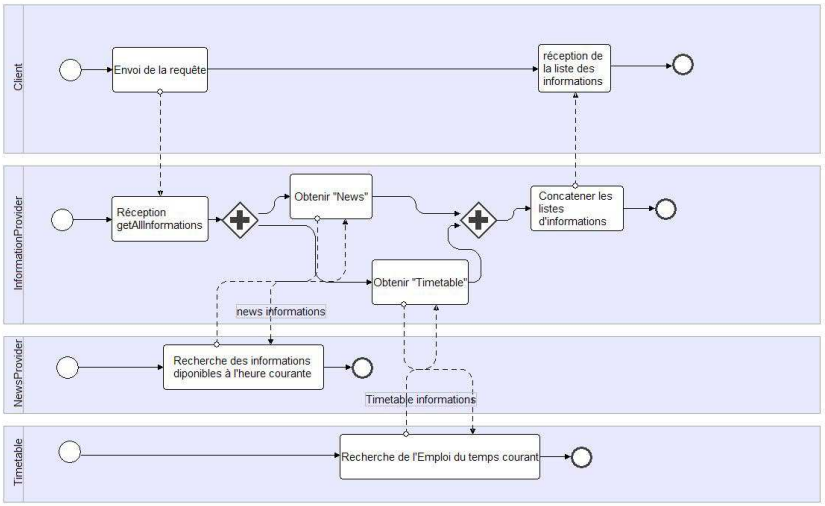
2.2 Contrats et Scénario d'adaptation

Les expérimentations menées dans Seduite présentées ici se décomposent en deux parties majeures : *(i)* la surveillance des temps de réponse de l'application initiale et *(ii)* la modification des assemblages de services pour tenir compte de l'arrivée d'une nouvelle source d'informations.

[3]Business Process Modeling Notation (BPMN) est une notation graphique pour modéliser des processus métier dans un "workflow".



(a) Services de SEDUITE à l'EPU Polytech'Sophia (EPUB)



(b) Orchestration Initiale "InformationProvider"

FIG. 1 – *Apercu général du système*

Aspect contractuel : Rapidité de diffusion La diffusion des informations nécessite évidemment de donner les informations dans un temps «*raisonnable*», les personnes n'acceptant pas de rester trop longtemps devant les écrans. Nous limitons le respect de ces délais à certaines heures de la journée, afin d'autoriser des phases de maintenance.

Informellement, cette obligation peut s'énoncer ainsi[4] :

> *L'exécution de l'opération o du Web Service S effectuée entre le temps t_i et t_j doit renvoyer son résultat en respectant le délai $\Delta_{S(o)}(t_i, t_j)$.*

Nous assujettissons donc les opérations concernées des services `Timetable` et `NewsProvider` à respecter respectivement les délais $\Delta_{TimeTable}$ et Δ_{News}.

L'opération *GetAllInformations* du service `InformationProvider` correspondant à l'orchestration de ces services sources peut donc s'engager à respecter un délai : $max(\Delta_{TimeTable}, \Delta_{News}) + \Delta_{cons}$ où Δ_{cons} correspond au temps estimé de concaténation et transformation des informations obtenues des deux précédents services.

Aspect contractuel : Unicité de domaine pour les sources d'information Plusieurs groupes d'étudiants ont travaillé sur le système de diffusion des informations. Afin de contrôler les modifications dynamiques d'assemblages de service, nous avons choisi d'interdire l'ajout de sources d'informations mettant en jeu des Web Services dont le domaine n'est pas celui de `InformationProvider`. L'opération de modification dynamique des orchestrations du service `OrchestrationServer` est donc protégé par l'obligation suivante pour le client :

> *L'ajout d'une source d'informations $source$ au service $InformationProvider$ n'est possible que si la nouvelle source est dans son domaine*[5] *:* $source.domain = InformationProvider.domain$

Ajout d'un nouveau service : Impacts sur les assemblages et les contrats Lorsque le service des emplois du temps des enseignants (`WebCal`) devient accessible, l'orchestration initiale `InfoProvider` est étendue pour prendre en compte cette nouvelle source d'information. Le contrat sur la rapidité de diffusion établi précédemment sur l'orchestration doit alors *(i)* soit être modifié pour l'adapter au nouvel assemblage et inclure les temps d'accès à cette nouvelle source d'information :$max(\Delta_{TimeTable}, \Delta_{News}, \Delta_{WebCal}) + \Delta_{cons^*}$, *(ii)* soit interdire cet ajout si le Δ_{webcal} est supérieur à celui annoncé par l'`InfoProvider`. Nous discutons ce choix de politique en section 4.

Sur la base de ce scénario simplifié, nous étudions à présent une mise en œuvre de l'application (cf. section 3) qui intègre les contrats et les évolutions dynamiques d'assemblages. Nous montrons les difficultés pour prendre en compte, sans modifier la plate-forme, les ajustements nécessaires des contrats relativement aux modifications d'assemblage.

3 Evolution dynamique : Web Services et contrats

La validation de contrats n'est pas aujourd'hui supportée par les plates-formes industrielles serveurs de web services. La modification dynamique des obligations et donc les violations de

[4] Une telle obligation est un exemple classique dans la littérature des contrats. Dans le cadre du projet RNTL FAROS dans lequel s'inscrivent ces travaux, on en trouve une définition similaire dans (Le Meur et al., 2007).

[5] Espace de nommage par exemple, pour garantir une cohérence au sein de la SOA.

contrats ne sont donc pas non plus gérées par ces plates-formes. La modification dynamique des orchestrations n'est pas non plus supportée par les moteurs d'exécution d'orchestrations.

Dans ce contexte, différents travaux académiques proposent des plates-formes prenant en charge l'évolution dynamique des applications. Dans le cadre du projet FAROS, cinq plates-formes sont envisagées ; leurs mécanismes d'exécution des contrats sont décrits dans le livrable (Balligand et al., 2007). La complexité de l'introduction dynamique des obligations et la validation de contrats est trop importante pour être abordée exhaustivement dans cet article. Plus d'informations sur ces plates-formes et leur capacité à prendre en charge les contrats est disponible dans le document précédemment cité. L'expérience relatée dans cet article repose sur la plate-forme ADORE/COCONET qui se situe dans la continuité de travaux antérieurs (CONFRACT pour les contrats (Collet et al., 2005) et NOAH pour les interactions entre composants (Blay-Fornarino et al., 2004)). Cette plate-forme vise à adapter les travaux précédents relatifs aux composants aux principes des architectures à base de Web Services.

3.1 La plate-forme ADORE/COCONET

Elle résulte de la combinaison de deux infrastructures, l'une COCONET dédiée à la surveillance dynamique de Web Services, l'autre ADORE permettant la modification dynamique d'assemblages de Web Services.

COCONET La plate-forme COCONET[6] permet la contractualisation d'un ensemble de Web Services en interceptant les invocations émises par les clients. Des clauses de contrats s'expriment par des expressions régulières sur ces interceptions et correspondent à des vérifications diverses, notamment sous la forme de code correspondant à des assertions exécutables de type *pré/post*. Dans sa mise en œuvre actuelle, l'interception est assurée par la création d'un nouveau Web Service *proxy* intercalé entre le Web Service manipulé et ses clients. Cette technique est aussi utilisée dans d'autres plates-formes de surveillance dynamique de Web Services, certaines créant des proxies autour d'un moteur d'orchestration (Baresi et al., 2004; Baresi et Guinea, 2005), d'autres créant simplement une couche d'intercession pilotée par l'infrastructure de surveillance (Lazovik et al., 2004) comme dans COCONET. Contrairement à ces plates-formes, COCONET ne fournit pas de langage de spécification pour les contrats mais uniquement une infrastructure pour intercepter les appels sur les Web Services, en s'inspirant de l'organisation interne de la plate-forme CONFRACT (Collet et al., 2005, 2007b). En effet, les spécifications correspondantes seront, dans le cadre du procédé FAROS, exprimées au niveau des modèles, puis transformées vers du code compatible avec la plate-forme COCONET (cf. section 4).

ADORE La plate-forme ADORE[7] (Joffroy et al., 2007a) permet la modification dynamique du comportement d'opérations de Web Services par modification des assemblages. Basée sur des mécanismes d'interceptions identiques à ceux de COCONET et sur le paradigme *Orienté Aspects* (Douence, 2004), elle supporte les modifications dynamiques des opérations initialement existantes en ajoutant de nouvelles préoccupations. Ces dernières peuvent aussi bien

[6] *COntrats sur COmposants .NET*
[7] *Aspect & Distributed ORchEstrations*

être des préoccupations techniques (sécurité, trace, ...) que des préoccupations métier (délégation du comportement à un nouveau service, modification dynamique d'une orchestration, ...) (Joffroy et al., 2007b).

3.2 Mise en œuvre des contrats dans CoCoNet

Contrat simple de contrôle des délais La mise en œuvre du contrat visant à assurer le respect des délais repose sur le déclenchement d'une violation lorsque le temps de réponse est trop important. Une entité *stratégie* nous permet de moduler les politiques de réaction en cas de violation de contrats. Le listing 1 présente la définition de ce contrat dans C# à l'aide de l'API CoCoNet.

```
namespace TimeScript
{
  public class TimeContract
  {
    private static Dictionary <int,Stopwatch> _timers;

    // Code pre-appel : lancement d un chronometre
    public class PreCallHandler : Observer
    {
      public override bool callHandler(CallEvent callEvent)
      {
        Stopwatch timer = new Stopwatch();
        lock (_timers) {_timers.Add(callEvent.CallId,timer);}
        timer.Start();
        return true;
      }
    }

    // Code post-appel : calcul du temps et reaction
    public class PostCallHandler : Observer
    {
      public override bool callHandler(CallEvent callEvent)
      {
        Stopwatch timer = null;
        lock (_timers) {
          timer = _timers[callEvent.CallId];
          _timers.Remove(callEvent.CallId);
        }
        timer.Stop();
        TimeSpan time = timer.Elapsed;
        if (time.Seconds > Settings.GetMaxTime(CallEvent)) {
          return ViolationStrategy.getDefaultStrategy().execute(null);
        }
        return true;
      }
    }
  }
}
```

Listing 1 – *Contractualisation CoCoNet du temps de réponse*

Contrat composite de contrôle des délais La mise en oeuvre de ce contrat repose sur l'utilisation de l'entité `Settings` qui à partir de la nature de l'événement détermine la valeur du Δ autorisé. La composition des contrats est donc prise en charge directement par cette entité. Cette solution masque la composition des contrats, rendant d'autant plus difficile sa maintenance. Une réification des contrats (*à la* CONFRACT par exemple) est donc nécessaire à la mise en œuvre d'une composition efficace et automatique des différents contrats mis en jeu. Certains éléments de modèle conçus à cet effet seront présentés dans la section 4.2.

Contrôle sur les ajouts de partenaires Ce contrat est placé en entrée du service `OrchestrationServer` et permet de vérifier que les nouvelles sources d'informations font partie du même domaine que celui de l'orchestration. En cas de violation, l'action n'est pas effectuée. Le listing 2 montre le code COCONET permettant l'écriture d'une telle obligation.

```
namespace DomainContract
{

  public class PreCallHandler: Observer
  {
    public override bool callHandler(CallEvent callEvent)
    {
      AbstractOrchestration _ao =
    Xml.Convert(this.Parameters[0]) as AbstractOrchestration ;
      Binding _bind = Xml.Convert(this.Parameters[1]) as Binding;

      // est-ce le schema d ajout de source ?
      if ( _ao.Type != "AddSource")
    return true; // On en poursuit pas la verification
      // Notre InfoProvider est il la cible ?
      if ( _bind.Services[_ao.Rules[0].Target].Url !=
           "http://.../InfoProvider.asmx")
    return true; // idem
      // Les services sont ils dans le bon namespace ?
      foreach(Service _s in _bind.Services) {
        if (_s.Domain != "rainbow.essi.fr/epub")
          return ViolationStrategy.GetDefaultStrategy().Execute(null);
      }
      return true;
    }
  }
}
```

Listing 2 – *Contractualisation CoCoNet du domaine des sources*

La figure 2 visualise une violation en cascade détectée par la plate-forme. La violation du temps de réponse au niveau des *sources* d'information entraînent une violation de l'orchestration composite les agrégeant.

Modification dynamique de l'assemblage Afin d'ajouter le service de calendrier partagé `WebCal`, les mécanismes de la plate-forme ADORE sont utilisés pour modifier dynamiquement l'orchestration `InformationProvider`. Si cette évolution dynamique est rendue aisée par ces mécanismes, l'absence de réification du contenu des contrats ne nous permet pas

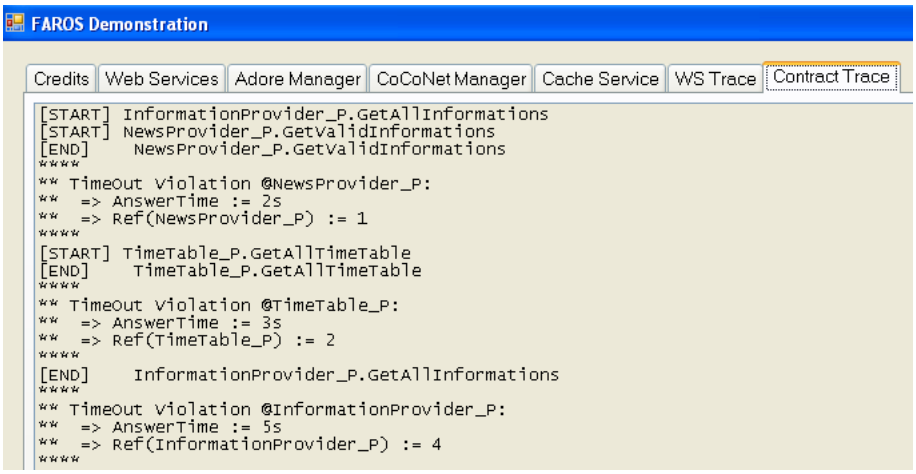


FIG. 2 – *Detection de la triple violation dans le démonstrateur*

d'automatiquement mettre à jour le contrat sur l'orchestration ou de notifier éventuellement d'une rupture de contrat en ajoutant cette source d'information.

Alors qu'à la conception du système, la corrélation entre les contrats et les assemblages a été facilement explicitée, sa prise en charge au niveau de la plate-forme en réaction à des modifications d'assemblage apparaît plus problématique. Bien sûr d'autres choix de plates-formes aurait pu simplifier certains éléments mais le problème central de la détermination des impacts entre les contrats et les assemblages reste valide. Nous montrons à présent comment nous envisageons d'aborder ce problème.

4 De la nécessité de l'ingénierie des modèles

Le travail présenté dans cet article se positionne dans le cadre du projet RNTL FAROS, qui a pour objectif de définir un environnement de composition pour la construction fiable d'architectures orientées services. Nous décrivons ci-après ce procédé au travers de notre scénario, puis nous montrons comment les modèles de contrats peuvent être utilisés pour contrôler les modifications d'assemblages de services.

4.1 Du modèle au code

Nous nous situons dans un processus classique d'IDM, en considérant une modélisation de l'application et son raffinement jusqu'à obtenir le code de l'application au niveau de la plate-forme.

La description donnée au niveau du scénario nous situe au niveau de l'application déployée à l'EPU Polytech'Nice-Sophia. Une autre application est déployée dans un Institut d'Education Sensoriel et est adaptée aux enfants handicapés. Cette dernière fait l'usage d'autres sources d'informations et d'autres orchestrations. Néanmoins les exigences et la distinction entre les

services dits sources d'informations ou techniques restent valides. Un travail est en cours pour affiner ces concepts et exigences au niveau d'un métamodèle et pour les utiliser au niveau des applications. L'objectif est de constituer un métamodèle métier qui facilite la conception des applications, favorise la vérification de contraintes et facilite l'expression des contrats.

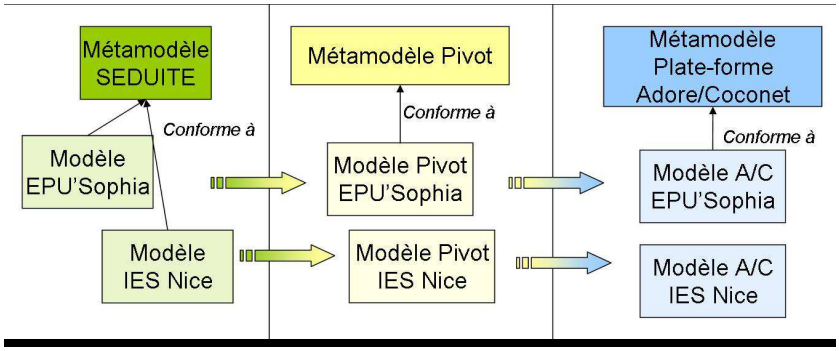


FIG. 3 – *Application du procédé* FAROS *sur* SEDUITE

Plusieurs plates-formes doivent servir de supports à de telles applications. Dans le contexte de SEDUITE, les supports de diffusion sont pris en charge par la plate-forme WCOMP (Cheung-Foo-Wo et al., 2006) qui permet des adaptations dynamiques de composants embarqués. Il convient donc d'autoriser une définition *indépendante plate-forme* des services, orchestrations et contrats qui constituent l'application, tout en automatisant la production des codes.

Pour prendre en charge l'hétérogénéité des plates-formes, nous défendons l'idée qu'un métamodèle pivot des services et contrats est une solution. Celui-ci devient alors la cible des transformations en provenance des modèles métiers. A partir des modèles obtenus, plusieurs plates-formes peuvent alors être ciblées. Le métamodèle pivot a été défini dans le rapport (Plouzeau et al., 2007), et est en cours de raffinement via les différentes expérimentations menées tant au niveau des plates-formes que des différentes applications cibles. La figure 3 schématise ce procédé.

4.2 Définition et mise en œuvre des contrats

Au niveau du métamodèle pivot, la représentation des contrats réutilise quasiment entièrement celle utilisée par le système ConFract[8]. La figure 4 en donne un extrait, un contrat étant constitué de clauses, chacune relative à une spécification organisée selon le paradigme *hypothèse-garantie* (Abadi et Lamport, 1993). Chaque clause référence un élément de l'architecture contractualisée qui est responsable de la véracité de la spécification. Un contrat peut aussi réifié un accord entre les parties relatives aux clauses, mais ceci n'est pas utilisé dans notre étude.

A partir du modèle pivot, le procédé FAROS prévoit ainsi de générer les formes de contrat adaptées au type de contrat et à la plate-forme ciblée. Ainsi le contrat de temps de réponse, correspondant au code du listing 2, sera représenté dans le modèle de plate-forme propre à

[8]Le lecteur trouvera le détail de ce modèle dans (Collet et al., 2007b). Il constitue la partie centrale de l'évolution du système ConFract, appelé Interact.

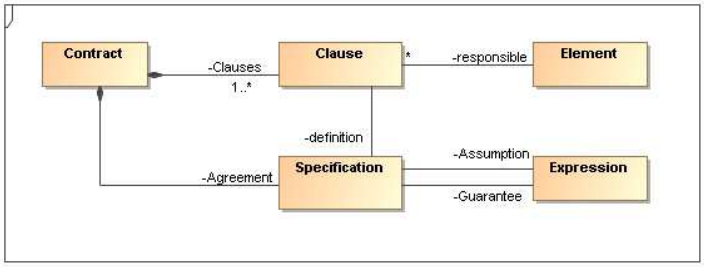


FIG. 4 – *Métamodèle de contrats (ConFract/FAROS).*

Adore/CocoNet par le modèle de la figure 5. On y retrouve des éléments contractuels communs (CONTRACT en TIMECONTRACT, CLAUSE en OPERATIONTIMINGCLAUSE, ELEMENT en WEBSERVICE). Comme les spécifications ne sont pas réifiées dans le code CocoNet résultant, certains éléments, comme SPECIFICATION et EXPRESSION, n'apparaissent plus. Le modèle est aussi complété par les types d'événement et les événements propres à la plate-forme qui vont permettre la vérification dynamique de ce type de contrat par la production du code équivalent au listing 2.

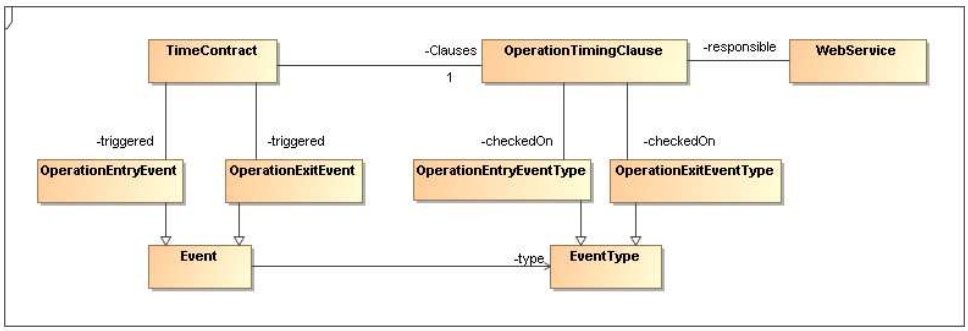


FIG. 5 – *Modèle du contrat de temps de réponse spécifique à* COCONET.

4.3 Contrôle des modifications d'assemblage guidé par les modèles

Nous nous proposons maintenant d'enrichir les mécanismes de production présentés précédemment pour contrôler les modifications d'assemblages des Web Services. Nous avons vu comment les spécifications des orchestrations et des différents contrats peuvent être exploitées pour surveiller et étendre le processus de récupération des informations au sein de l'application. Les actions d'ajout de contrats et de modification des assemblages ont été jusqu'à présent dirigées par l'application, et en l'occurrence le programmeur. Nous évaluons à présent différentes stratégies pour contraindre ou, au contraire, relâcher le processus de diffusion des informations

au sein de l'application, en fonction des modifications d'assemblages.

Différentes stratégies sont ainsi envisageables :

- Si les contraintes attendues du service *InformationProvider* sont imposées et non déduites de l'orchestration, il convient de déceler à l'extension de l'orchestration, le problème du à l'ajout d'un service ne permettant plus de respecter les obligations initiales. Lorsque le service ajouté ne spécifie pas d'obligation, son ajout peut être refusé. Nous disposons des briques logicielles pour tester et contrôler les extensions (détection de l'extension d'orchestration, accès aux temps maximum imposés, ...), néanmoins la détermination même de la violation et son calcul sont bien plus faciles à définir au niveau du modèle que de la mise en oeuvre.
- La partie obligation exposée par le service *InformationProvider* correspond à une fonction calculée à partir des temps maximums offerts par les services sources. Elle peut alors être directement déduite de la modélisation. Cette fois-ci, il est encore bien plus intéressant d'opérer ces calculs au niveau du modèle, et bien sûr de les définir au niveau du métamodèle.
  Si la spécification du contrat au niveau du modèle est relativement simple, ce n'est pas le cas au niveau de l'implémentation où les objets à manipuler sont propres à la plate-forme cible et aux choix d'implémentation.
  La mise en oeuvre du contrat spécifié comme forçant l'obligation de rendre le service en un temps inférieur au maximum des temps accordés aux services composants doit prendre en compte les modifications potentielles d'assemblages. Si nous manipulons les modèles à l'exécution, le contrat peut être judicieusement adapté et regénéré. Inversement, il devra se baser sur une introspection ou être considéré comme obsolète en cas d'évolution de l'architecture. Encore une fois, en présence d'une évolution dynamique des applications, il convient d'établir au niveau de la modélisation le comportement futur du système.

5 Conclusion et perspectives

Dans cet article, nous avons présenté la démarche et les premières mises en œuvres expérimentales pour la prise en compte de contrats dans des architectures orientées services, en suivant une approche dirigée par les modèles. Ces premiers résultats permettent de déterminer les transformations nécessaires au passage d'exigences définies au niveau des modèles métiers vers une plate-forme qui supporte l'évolution dynamique d'architectures orientées services.

Ces travaux sont effectués dans le cadre du projet RNTL FAROS, dont le procédé s'appuie sur un métamodèle pivot des services et des contrats. A partir d'une modélisation métier propre à une application en exploitation de diffusion d'information, nous avons établi plusieurs contrats et scénarios d'adaptation. Nous avons détaillé leur mise en œuvre jusqu'au code et montré comment la notion de contrat est réifiée et manipulé tout au long du procédé d'ingénierie des modèles. La présentation d'une seule projection vers une plate-forme a été motivée par une volonté de concision, mais un des livrables du projet (F-3.1) (Balligand et al., 2007) présente les différentes implémentations de ce contrat dans les plates-formes du projet à savoir : CONFRACT(Collet et al., 2005), AOKELL(Seinturier et al., 2006), WCOMP(Cheung-Foo-Wo et al., 2006), FAC(Pessemier et al., 2006), ORQOS(Baligand et al., 2007).

La définition de contrats et la composition des services sont fortement corrélées. L'IDM nous paraît une bonne manière de procéder au calcul automatique des compositions de contrats. Ce point est d'autant plus important que les applications orientées services sont particulièrement sujettes aux évolutions dynamiques et que plusieurs des plates-formes de validation du projet offrent cette capacité. L'expression et le calcul au niveau de chaque plate-forme de la réaction aux adaptations est complexe, et suppose d'éventuelles extensions des plates-formes. Or, il est évident que nous ne pouvons anticiper les mécanismes qu'offriront les plates-formes industrielles de demain. Au contraire, une approche au niveau des modèles devrait nous permettre de supporter des applications multi-plateformes, tout en modularisant les calculs nécessaires et en réutilisant probablement à terme les résultats obtenus dans le domaine formel. Il est cependant à noter que les domaines applicatifs qui nous intéressent autorisent des temps de latence suffisant pour envisager ces changements de niveaux.

Remerciements

Ce travail a été financé par l'ANR au travers du projet RNTL FAROS. Nous remercions l'ensemble des membres du projet RNTL FAROS avec qui la démarche a été élaborée au travers de discussions et de développements fort enrichissants.

Références

Abadi, M. et L. Lamport (1993). Composing specifications. *ACM Trans. Program. Lang. Syst. 15*(1), 73–132.

Baligand, F., N. Rivierre, et T. Ledoux (2007). A declarative approach for qos-aware web service compositions. In B. J. Krämer, K.-J. Lin, et P. Narasimhan (Eds.), *ICSOC*, Volume 4749 of *Lecture Notes in Computer Science*, pp. 422–428. Springer.

Balligand, F., M. Blay-Fornarino, H. Chang, D. Cheung-Foo-Wo, P. Collet, G. Dufrêne, V. Hourdin, S. Lavirotte, S. Mosser, A. Ozanne, A.-M. Pinna-Déry, N. Rivierre, L. Seinturier, et J.-Y. Tigli (2007). Identification des modalités de prise en charge des contrats pour chaque plate-forme cible. Technical Report F.3.1, RNTL Faros.

Baresi, L., C. Ghezzi, et S. Guinea (2004). Smart Monitors for Composed Services. In *ICSOC'2004*. ACM.

Baresi, L. et S. Guinea (2005). Towards Dynamic Monitoring of WS-BPEL Processes. In *ICSOC'2005*, Springer - LNCS. Springer LNCS.

Blay-Fornarino, M., A. Charfi, D. Emsellem, A.-M. Pinn a Déry, et M. Riveill (2004). Software interaction. *Journal of Object Technology (ETH Zurich) 3*(10), 161–180.

Blay-Fornarino, M., P. Collet, P. Lahire, S. Lavirotte, A.-M. Pinna-Déry, et J.-Y. Tigli (2007). Contrats et compositions de services de l'application seduite (services de diffusion ubiquitaire d'informations dans des etablissements scolaires). Technical Report F.4.1, RNTL Faros.

Cheung-Foo-Wo, D., J.-Y. Tigli, S. Lavirotte, et M. Riveill (2006). Wcomp : a Multi-Design Approach for Prototyping Applications using Heterogeneous Resources. In *17th IEEE International Workshop on Rapid System Prototyping (RSP)*, Chania, Crete.

Collet, P., T. Coupaye, H. Chang, L. Seinturier, et G. Dufrêne (2007a). Components and Services : A Marriage of Reason. Research Report I3S-RR2007-17FR, I3S Lab, Sophia Antipolis, France.

Collet, P., J. Malenfant, A. Ozanne, et N. Rivierre (2007b). Composite Contract Enforcement in Hierarchical Component Systems. In *6th International Symposium on Software Composition (SC'07)*, Volume to appear of *LNCS*, Braga, Portugal. Springer Verlag.

Collet, P., R. Rousseau, T. Coupaye, et N. Rivierre (2005). A Contracting System for Hierarchical Components. In *Component-Based Software Engineering, 8th International Symposium (CBSE'2005)*, Volume 3489 of *LNCS*, St-Louis (Missouri), USA, pp. 187–202. Springer Verlag.

Douence, R. (2004). A restricted definition of AOP. In K. Gybels, S. Hanenberg, S. Herrmann, et J. Wloka (Eds.), *European Interactive Workshop on Aspects in Software (EIWAS)*.

Joffroy, C., S. Mosser, et M. Blay-Fornarino (2007a). plate-forme ADORE : Aspect and Distributed ORchEstrations. Technical report, I3S, Sophia-Antipolis (France).

Joffroy, C., S. Mosser, M. Blay-Fornarino, et C. Nemo (2007b). Des Orchestrations de Services Web aux Aspects. In *3ème Journée Francophone sur le Développement de Logiciels Par Aspects (JFLDPA'2007)*, Toulouse (France).

Lazovik, A., M. Aiello, et M. P. Papazoglou (2004). Associating assertions with business processes and monitoring their execution. In M. Aiello, M. Aoyama, F. Curbera, et M. P. Papazoglou (Eds.), *Service-Oriented Computing - ICSOC 2004, Second International Conference, New York, NY, USA, November 15-19, 2004, Proceedings*, pp. 94–104. ACM.

Le Meur, A.-F., P. Lahire, G. Waignier, P. Collet, N. Plouzeau, M. Dao, L. Duchien, A. Ozanne, et N. Rivierre (2007). Spécification d'une architecture pour la contractualisation de services : expression du besoin. Technical Report F-1.2., RNTL Faros.

Ludwig, H., A. Dan, et R. Kearney (2004). Cremona : An Architecture and Library for Creation and Monitoring of WS-Agreements. In *ICSOC'2004*. ACM.

MacKenzie, M., K. Laskey, F. McCabe, P. Brown, et R. Metz (2006). Reference Model for Service Oriented Architecture 1.0. Technical Report wd-soa-rm-cd1, OASIS.

Open SOA (2007). *SCA Service Component Architecture - Assembly Model Specification*. Open SOA. Version 1.00.

Pessemier, N., L. Seinturier, L. Duchien, et T. Coupaye (2006). A model for developing component-based and aspect-oriented systems. In *Proceedings of the 5th International Symposium on Software Composition (SC'06)*, Volume 4089 of *Lecture Notes in Computer Science*. Springer.

Plouzeau, N., F. Chauvel, et G. Waignier (2007). Spécification du méta-modèle pivot. Technical Report F2.1, RNTL FAROS.

Seinturier, L., N. Pessemier, L. Duchien, et T. Coupaye (2006). A Component Model Engineered with Components and Aspects. In *Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE'06)*, Volume 4063 of *Lecture Notes in Computer Science*, pp. 139–153. Springer.

White, S. A. (2006). *Business Process Modeling Notation (BPMN)*. IBM Corp.

Zou, Z. et Z. Duan (2006). Building Business Processes or Assembling Service Components : Reuse Services with BPEL4WS and SCA. In *ECOWS'2006*. IEEE Computer Society Press.

Summary

Services Oriented Architectures (SOA) provides flexibility but intensifies the need in contract definition for services. Contracts expression must then be included at all levels of the software lifecycle, entailing different implementations according to their nature and the targeted execution platforms. In the national research project Faros, experiments have been conducted to introduce contract support in an SOA-based application. This paper presents the resulting process and the implementations. It also discusses benefits of Model Driven Engineering applied to evolution management of the application.